Expansion of MARTA to Local Suburbs Through Optimized Combinatorial Algorithms

Mitesh Shah, Sohaib Samad, Sanjar Zaman, Branden Kim

MATH 3012-QHS: Applied Combinatorics

Dr. Yaofeng Su

December 1, 2023

Abstract

MARTA currently only provides transportation to downtown Atlanta, leaving out highly populated areas of Metro Atlanta like the suburbs. The aim of this project is to first expand MARTA by generating several new stations based on population density maps and then create optimized routes between all vertices, given the requirement of being an Eulerian circuit. 122 additional MARTA stations were added to generate a fuller web. An overarching Eulerian circuit for the graph as a whole was generated to maximize efficiency, using a hybrid of a K-Nearest Neighbors model with Dijikstra's algorithm to find the optimal circuit with shortest paths between stations. By targeting the high-population-density locations, a successful Eulerian circuit was developed, creating a feasible route that maximizes convenience to boost Metro Atlanta's accessibility for all.

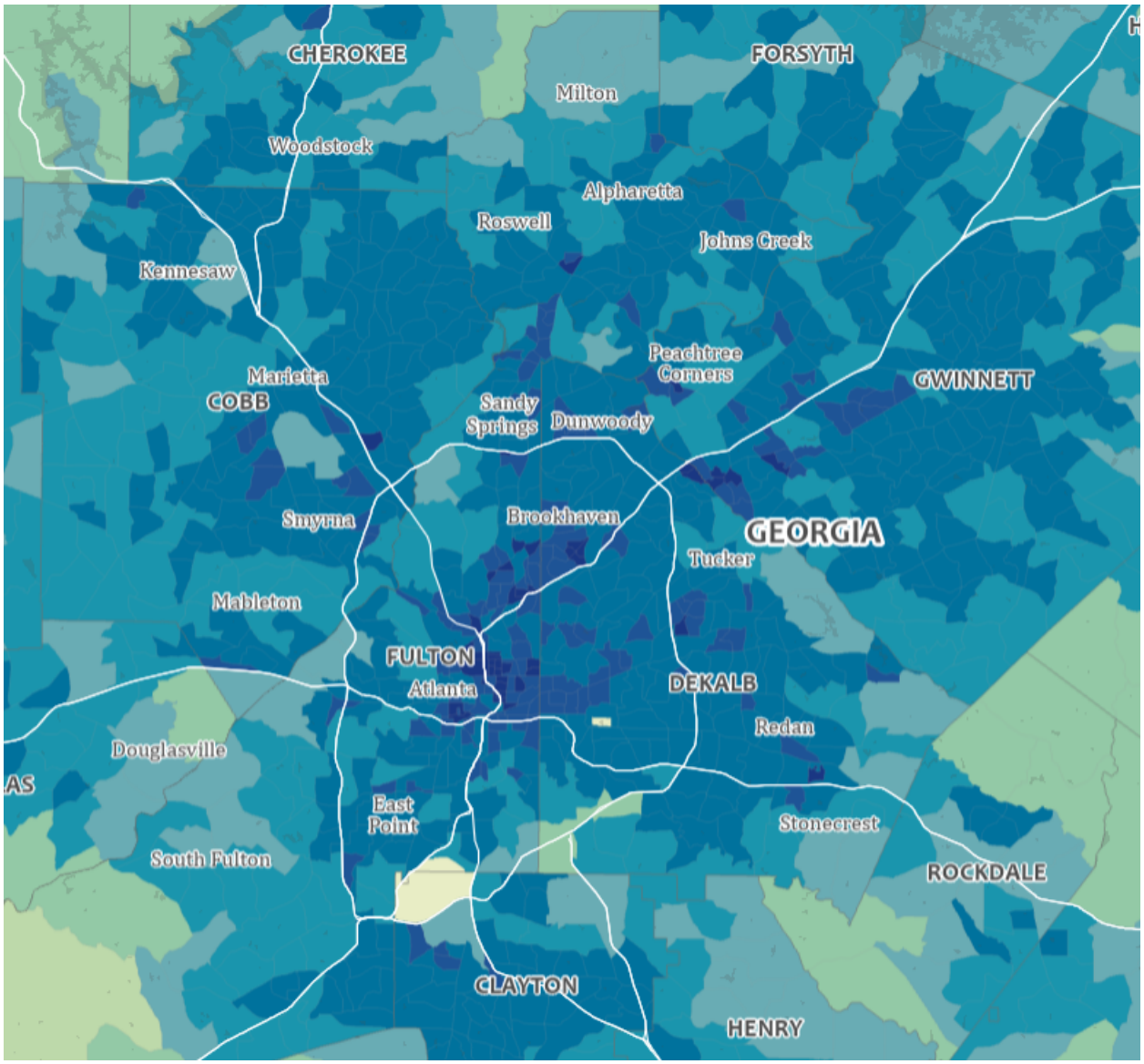
Background

MARTA is the premier subway system for Metro Atlanta; however, it does not currently extend into the suburban areas surrounding Atlanta. Expanding public transportation into suburban areas can solve issues like congestion, carbon emissions, and social disparity. A recent INRIX study ranked Atlanta as the tenth most congested urban area in the United States, resulting in a $1,257 cost per driver and a $3.1 billion cost for the city (Inrix, 2023). Traffic congestion can be easily relieved by optimizing and expanding public transportation since that will help free up the roads and offer alternative modes of transportation. By accomplishing this, residents and the city can save money, which can be invested into other programs rather than dealing with worsening congestion issues. For suburban counties, such as Gwinnett County, emissions from transportation significantly exceed emissions from other sectors, alone tripling the second-largest sector (Drawdown, 2023). By expanding MARTA, the need for individual

transportation can be greatly reduced with the onset of more widespread and accessible public transportation. Not only will this expansion help to alleviate tensions among residents, the city, and the globe, but it will also help to solve societal issues. By granting everybody equal access to transportation across Metro Atlanta, more residents can commute to their jobs and other places of importance, even without personal transportation. This can, in turn, better address the socioeconomic gap between classes while combating regionalized discrimination.

To alleviate these current problems in MARTA, the proposed solution is to expand the current train system in Metro Atlanta. To combinatorially optimize MARTA routes for the expanded system, an algorithm made up of a combination of the machine learning model K-Nearest Neighbors and Dijkstra's algorithm is created. The K-Nearest Neighbors model is a supervised learning classifier that essentially allows for groupings of the data points to be made based on their proximity to one another, discovering similarities across multiple classifications. This model goes hand-in-hand with Dijkstra's algorithm, which, according to Keller & Trotter (2017) searches for the shortest path between two nodes. By applying a K-Nearest neighbors model with Dijkstra's algorithm, the shortest pathways between a starting and ending location can be discovered and through multiple phases of training, develop a high level of confidence in the generated graphs. The basis of the pathways are Eulerian circuits, as Eulerian paths travel along each edge only once, reducing redundancy (Keller & Trotter, 2017). By applying this multifaceted model, an optimal and cost-minimizing solution for MARTA's expansion can be discovered.

Methods



*A population density map for the Metro-Atlanta area, deeper shades of blue are more densely populated (*2020 Census Demographic Data Map Viewer, n.d.).

This map was utilized to find the most ideal locations for new stations. More densely populated areas demand more stations to satisfy the transportation needs of the local population. After finding the most populated regions, a novel K-Nearest Neighbors algorithm using a distance function was established (Hu et al., 2016). In the algorithm, each vertex could have a

maximum of between one and twenty edges to prevent congestion of railways. The edges generated via the K-Nearest Neighbors algorithm were weighted by distance, allowing for only the shortest edges to be considered. Using Euclidean distance, the initial set of railways to connect stations was generated. After establishing this initial network, observational analysis was employed to remove redundant edges, thus reducing complexity. This process computes with a time complexity of O(nk), where n and k represent the number of stations and number of edges out of each station.

$$d = |\mathbf{x} - \mathbf{y}| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$

Euclidean distance formula.

Following this process, an Eulerian function was applied to produce the graph and its Eulerian edges, allowing us to discover the Eulerian circuit. Alongside this was an implementation of Dijkstra's algorithm for efficient pathfinding between any two stations on the map. Using the distance-weighted edges and a Greedy-based algorithm, the program takes in two user inputs (starting location and destination) and returns the shortest path from point A to point B.

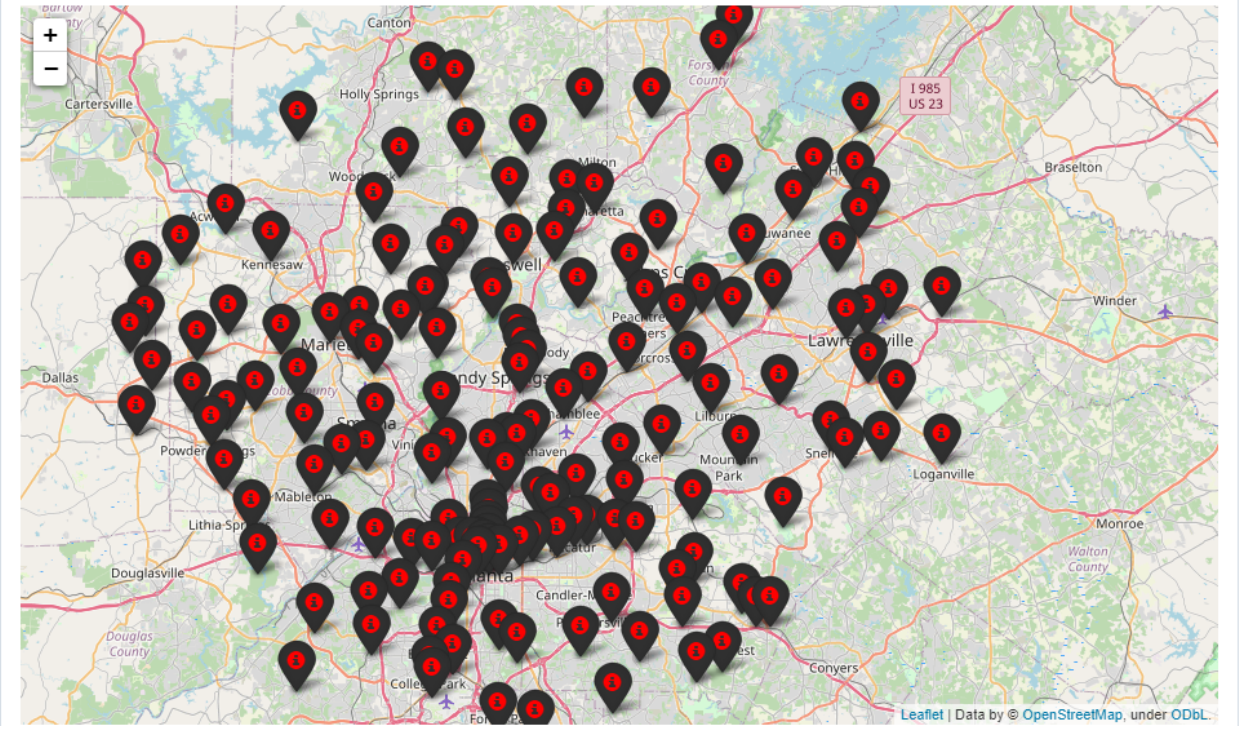
```python
class Graph():

    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]

    def printSolution(self, dist):
        print("Vertex \t Distance from Source")
        for node in range(self.V):
            print(node, "\t\t", dist[node])

    # A utility function to find the vertex with minimum distance value
    def minDistance(self, dist, sptSet):

        # Initialize minimum distance for next node
        min = 1e7

        for v in range(self.V):
            if dist[v] < min and sptSet[v] == False:
                min = dist[v]
                min_index = v

        return min_index

    # Function that implements Dijkstra's single source shortest path algorithm
    def dijkstra(self, src):

        dist = [1e7] * self.V
        dist[src] = 0
        sptSet = [False] * self.V

        for cout in range(self.V):

            # Pick the minimum distance vertex from the set of vertices not yet processed.
            # u is always equal to src in first iteration
            u = self.minDistance(dist, sptSet)

            # Put the minimum distance vertex in the shortest path
            sptSet[u] = True

            # Update dist value of the adjacent vertices
            # of the picked vertex only if the current
            # distance is greater than new distance and
            # the vertex in not in the shortest path tree
            for v in range(self.V):
                if (self.graph[u][v] > 0 and
                   sptSet[v] == False and
                   dist[v] > dist[u] + self.graph[u][v]):
                    dist[v] = dist[u] + self.graph[u][v]

        self.printSolution(dist)
```

Example function that conducts Dijkstra's algorithm (not implemented in project as we used NetworkX, this is purely for mathematical clarity).

Lastly, a feed-forward graphical neural network was created to create node embeddings. These node embeddings are vector representations of the nodes in the graph, each of which contain identifiable markers that can be used by passengers to optimize transfers, routing, and updates through updated encoded information about station connectivities, delays, crowding, and accessibility (Dehghan-Kooshkghazi, 2022).

Results

```
import folium

mapmarta=folium.Map(location=[33.934150, -84.246706])

for ind in range(len(martabase["Station"])):
    name = martabase.iloc[ind,1]
    lat = martabase.iloc[ind,2]
    longi = martabase.iloc[ind,3]
    folium.Marker([lat,longi],tooltip=name, icon=folium.Icon(color='black',icon_color='red')).add_to(mapmarta)
mapmarta
```



Folium program that prints a map of the new MARTA stations with a total of 160 stations (all vertices on the map represent stations—both old and new).

This map presents our dataset of stations: the most populated areas in the Metro Atlanta area for efficient transportation. With these locations, most residents in Metro Atlanta will have access to MARTA subways and be able to get from one location to another without having to make excessive stops. Certain locations were chosen for their popularity and significance, like local parks or malls.

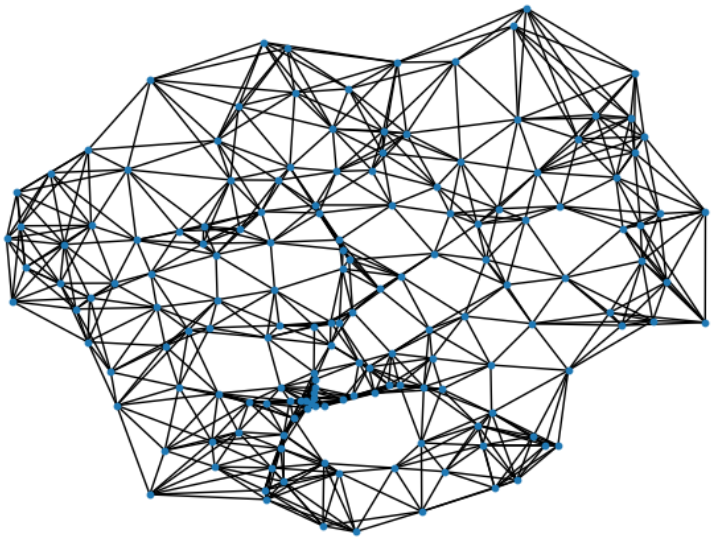
Any new stations are regional approximations, not precise locations for new stations. Approximations allow for focus exclusively on new and optimal routes without considering factors such as residential areas, privately owned land, and other logistical considerations. As such, economic factors like the cost of these new stations were also not considered.

```python
def distance_formula(x1, y1, x2, y2):
   dist = math.sqrt( (math.pow( (x1 - x2) ,2)) + (math.pow( (y1 - y2) ,2)) )
   return dist

def knn_edges(n, k, vertices):
    edges = []
    for key, v in vertices.items():
        # Values for x1,x2
        name = key
        y1 = v[0]
        x1 = v[1]

        # Calculate distances to other vertices
        distances = []
        for key2, u in vertices.items():
            if u[0]!=y1 and u[1]!=x1:
                try:
                    distances.append((u, distance_formula(x1, y1, u[1], u[0])))
                except Exception as e:
                    print(f"Error calculating distance: {e}")

        # Sort distances
        distances.sort(key=lambda x: x[1])

        # Select k nearest neighbors and add edges
        for j in range(k):
            u, _ = distances[j]
            val = {i for i in martagraph if martagraph[i]==u}
            va = val.pop()
            edges.append((name, va, distances[j][1]))

    return edges
```

```python
G = nx.Graph()

# Step 3: Add nodes and edges
for name, coordinates in martagraph.items():
    G.add_node(name, pos=coordinates)  # Add node with 'pos' attribute

edges = knn_edges(160, 8, martagraph)
#print(edges)
G.add_weighted_edges_from(edges)

# Plot the graph
pos = nx.get_node_attributes(G, 'pos')
nx.draw(G, pos, with_labels=False, node_size=10)
plt.show()
```

Graphing vertices and adding edges through KNN on graph

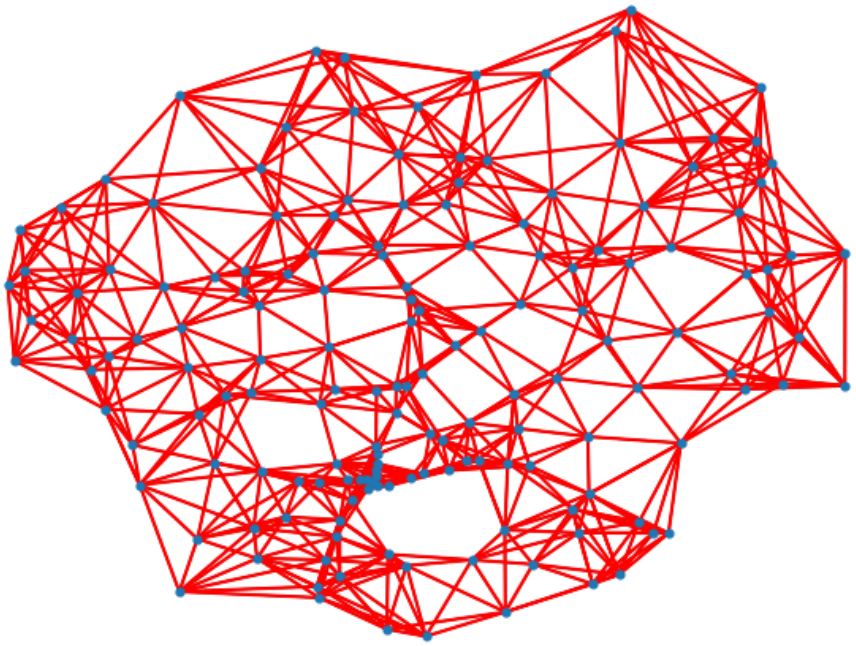
Then, each station is graphed using Networkx, a Python library that supports graph theory visualizations such as Dijstrka's Algorithm, with its coordinates serving as vertices. After that, the Euclidean distance formula is applied to each vertex against all other vertices to find the closest neighbors using K-Nearest-Neighbors approximations. Edges are created between each of these neighboring vertices, producing the graph that will be used for Euler circuits, pathfinding algorithms, and travel optimizations, as shown below.



Unlabeled graph with distance-weighted edges added

```
Ge = nx.euler.eulerize(G)

eulerian_circuit = list(nx.eulerian_circuit(Ge))

# Print the Eulerian circuit
print("Eulerian Circuit:", eulerian_circuit)

# Visualize the graph and the Eulerian circuit
pos = nx.get_node_attributes(Ge, 'pos')
nx.draw(Ge, pos, with_labels=False, node_size=12)
nx.draw_networkx_edges(Ge, pos, edgelist=eulerian_circuit, edge_color='r', width=1.5)
plt.show()
```

Eulerizing and computing Eulerian circuit for graph



Final Euler graph with Euler circuit embedded

```
# example path provided by app from Avalon to Airport station using weighted Dijkstra's algo
lp = True
while lp:
    try:
        start = input("Enter your starting location: ")
        end = input("Enter your destination: ")
        print(nx.dijkstra_path(G, start, end))
        lp = False
    except Exception as e:
            print(f"Error: {e}")
```

User input-based Dijkstra's/Greedy pathfinding algorithm

This text-based application allows a passenger to enter any two locations and receive the most optimized path weighted on both distance and travel time to get from one station to another using the graph, presenting the following route as the solution to get from Avalon to Airport

Station:

```
Enter your starting location: Avalon
Enter your destination: Airport
['Avalon', 'North Point', 'Horseshoe Bend ', 'North Springs', 'Medical Center', 'Buckhead', 'Lindbergh Center', 'Arts Ce
nter', 'Vine City', 'West End', 'Lakewood/Fort McPherson', 'Airport']
```

```python
import torch
import torch.nn as nn
import torch.optim as optim
import torch_geometric
import tensorflow as tf

node_features = torch.randn(160, 16)
data = torch_geometric.utils.from_networkx(G)
data.x = node_features
print(data)

# Define and train a GNN model
class GNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(GNN, self).__init__()
        self.conv1 = torch_geometric.nn.GraphConv(input_size, hidden_size)
        self.conv2 = torch_geometric.nn.GraphConv(hidden_size, output_size)

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = torch.relu(x)
        x = self.conv2(x, edge_index)
        return x

model = GNN(input_size=16, hidden_size=64, output_size=2)
optimizer = optim.Adam(model.parameters(), lr=0.01)
criterion = nn.CrossEntropyLoss()

# Train the model on graph data
for epoch in range(20):
    optimizer.zero_grad()
    output = model(data.x, data.edge_index)
    target = torch.ones_like(output)
    loss = criterion(output, target)
    loss.backward()
    optimizer.step()

    print(f'Epoch {epoch + 1}/{20}, Loss: {loss.item()}')

# uses the learned embeddings to guide Hamiltonian path search
node_embeddings = model.conv1(data.x, data.edge_index)
print(node_embeddings)
```

Graphical Neural Network (GNN) for Node Embeddings and applications

The novel GNN serves several key purposes for the user, making the new MARTA experience mathematically optimized and efficient for traveling at a low loss of approximately 1.7. The GNN consists of two graph convolutional layers, a Rectified Linear Unit (ReLU) activation function, an Adam optimizer, and a CrossEntropyLoss function, making it optimized for this set of parameters and node features.

Conclusions

After generating the optimized Eulerian circuit, each station is visited once, and a sufficient number of rails/connections are present such that complexity is reduced while increasing connectivity across Metro-Atlanta. The establishment of a novel network of MARTA stations and connection pathways allows for a more affordable, accessible transportation system to be present for all Georgia residents. While the political and social implementations of expanding MARTA have been widely considered, this study aims to provide a new perspective on the most efficient and socially optimal expansion of MARTA. This project, however, would require a substantial infrastructural investment. As such, revenue-generating methods and municipality budgets must be deeply considered and researched to implement the conclusions of this study. Furthermore, individual stations must be considered more carefully to ensure that local residents are neither displaced nor disturbed. Beyond the initial MARTA expansion, this system would be extended beyond just the highly populated regions, eventually creating a state-wide railway system to solve problems of congestion, carbon emissions, and social disparity across Georgia. The proposed system of railways in this study offers a feasible solution to guide the state of Georgia towards a more sustainable future.

References

2020 Census Demographic Data Map Viewer. (n.d.). https://maps.geo.census.gov/ddmv/map.html

Dehghan-Kooshkghazi, A., Kamiński, B., Kraiński, Ł., Prałat, P., & Théberge, F. (2022, August 4). *Evaluating node embeddings of complex networks*. Academic.oup.com. https://academic.oup.com/comnet/article-abstract/10/4/cnac030/6655565

Drawdown, G. (2023, August 14). *Drawdown Georgia GHG Emissions Tracker*. Drawdown Georgia. https://www.drawdownga.org/ghg-emissions-tracker/

Hu, L.-Y., Huang, M.-W., Ke, S.-W., & Tsai, C.-F. (2016, August 9). *The distance function effect on K-nearest neighbor classification for medical datasets - springerplus*. SpringerOpen. https://springerplus.springeropen.com/articles/10.1186/s40064-016-2941-7

Inrix, I. (2023, January 10). *INRIX: Return to Work, Higher Gas Prices & Inflation Drove Americans to Spend Hundreds More in Time and Money Commuting*. PR Newswire: press release distribution, targeting, monitoring and marketing. https://www.prnewswire.com/news-releases/inrix-return-to-work-higher-gas-prices--inflation-drove-americans-to-spend-hundreds-more-in-time-and-money-commuting-301717159.html

Keller, M. T., & Trotter, W. T. (2017). Applied combinatorics. Mitchel T. Keller, William T. Trotter

Appendix (Inputs and Respective Outputs)

Source code

12/1/23, 6:03 PM marta

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:
```
martabase = pd.read_csv("martabase_final.csv")
```

In [ ]:
```
martabase.head()
```

Out[ ]:

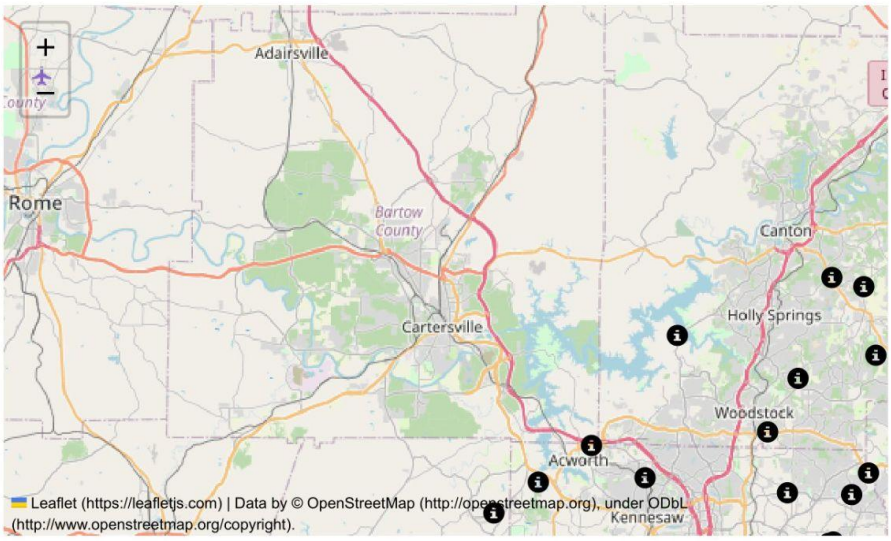
	ID	Station	Latitude	Longitude	Color
0	1	Airport	33.640758	-84.446341	NaN
1	2	Arts Center	33.789705	-84.387789	NaN
2	3	Ashby	33.756346	-84.417556	NaN
3	4	Avondale	33.775277	-84.281903	NaN
4	5	Bankhead	33.771890	-84.428840	NaN

In [ ]:
```
martabase.shape
```

Out[ ]:
```
(160, 5)
```

In [ ]:
```
import folium

mapmarta=folium.Map(location=[33.934150, -84.246706])

for ind in range(len(martabase["Station"])):
    name = martabase.iloc[ind,1]
    lat = martabase.iloc[ind,2]
    longi = martabase.iloc[ind,3]
    folium.Marker([lat,longi],tooltip=name, icon=folium.Icon(color='black',icon_col
mapmarta
```

Out[ ]:



In [ ]:

```
from collections import defaultdict

class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = defaultdict(list)

    def addEdge(self, v1, v2):
        self.graph[v1].append(v2)
        self.graph[v2].append(v1)

    def DepthFirst(self, v, visited_list):
        visited_list[v] = True
        for i in self.graph[v]:
            if visited_list[i] == False:
                self.DepthFirst(i, visited_list)

    def isConnected(self):
        visited_list = False*(self.V)
        for i in range(self.V):
            if(len(self.graph[i]) != 0):
                break
        if i == self.V-1:
            return True

        self.DepthFirst(i, visited_list)

        # Check if all non-zero degree vertices are visited
        for i in range(self.V):
            if visited_list[i] == False and len(self.graph[i]) > 0:
                return False

        return True
```

```
    def isEulerian(self):
        # Check if all non-zero degree vertices are connected
        if self.isConnected() == False:
            return 0
        else:
            # Count vertices with odd degree
            odd = 0
            for i in range(self.V):
                if len(self.graph[i]) % 2 != 0:
                    odd += 1
            #number of odds:
                # 0 -> euler cycle
                # 2 -> euler path
                # >2 -> not eulerian
            if odd == 0:
                return 2 #euler cycle
            elif odd == 2:
                return 1 #euler path
            elif odd > 2:
                return 0 #no euler

    def test(self):
        res = self.isEulerian()
        if res == 0:
            print("Graph is not Eulerian")
        elif res == 1:
            print("Graph has a Euler path")
        else:
            print("Graph has a Euler cycle")
```

In [ ]:
```
g = Graph(160)
g.addEdge(1,2)
```

In [ ]:
```
names = martabase.iloc[:,1]
latitudes = martabase.iloc[:,2]
longitudes = martabase.iloc[:,3]
longitudes.head()
```

Out[ ]:
```
0   -84.446341
1   -84.387789
2   -84.417556
3   -84.281903
4   -84.428840
Name: Longitude, dtype: float64
```

In [ ]:
```
npnames = names.to_numpy()
nplats = latitudes.to_numpy()
nplongs = longitudes.to_numpy()
```

In [ ]:
```
from sklearn.preprocessing import MinMaxScaler

nplats = nplats.reshape(-1, 1)
nplongs = nplongs.reshape(-1, 1)
```

```
lats_fin = MinMaxScaler().fit_transform(nplats)
longs_fin = MinMaxScaler().fit_transform(nplongs)
```

In [ ]:
```
lats_fin[0]
```

Out[ ]:
```
array([0.06162806])
```

In [ ]:
```
longs_fin[0]
```

Out[ ]:
```
array([0.37190522])
```

In [ ]:
```
longs = longs_fin.flatten()
lats = lats_fin.flatten()
```

In [ ]:
```
vertices_finnn = pd.DataFrame({'Station': npnames, 'Latitude': lats, 'Longitude': l

martagraph = vertices_finnn.set_index('Station')[['Longitude', 'Latitude']].apply(t
```

In [ ]:
```
vertices_f = martabase.iloc[:,2:4]
vertices_f.head()
```

Out[ ]:

	Latitude	Longitude
0	33.640758	-84.446341
1	33.789705	-84.387789
2	33.756346	-84.417556
3	33.775277	-84.281903
4	33.771890	-84.428840

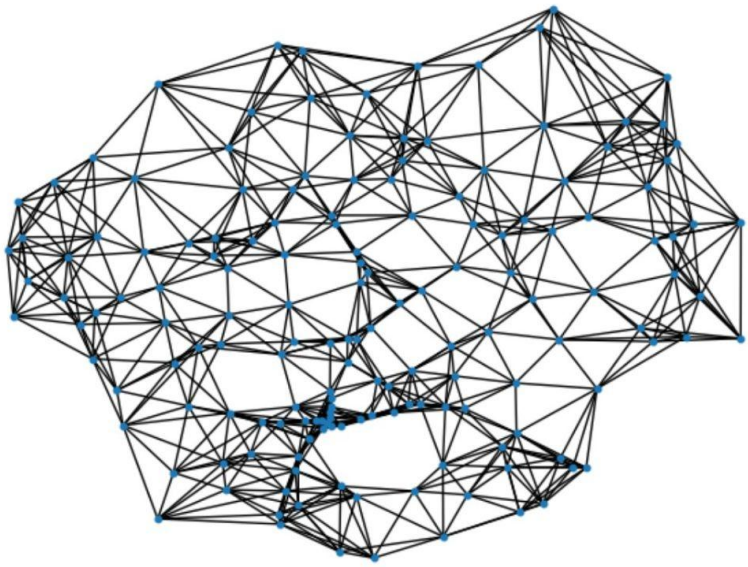
In [ ]:
```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import math
```

In [ ]:
```
vertices_final = vertices_f.to_numpy().tolist()
vertices_final[0]
```

Out[ ]:
```
[33.640758, -84.446341]
```

In [ ]:
```
def distance_formula(x1, y1, x2, y2):
   dist = math.sqrt( (math.pow( (x1 - x2) ,2)) + (math.pow( (y1 - y2) ,2)) )
   return dist

def knn_edges(n, k, vertices):
    edges = []
    for key, v in vertices.items():
        # Values for x1,x2
        name = key
        y1 = v[0]
        x1 = v[1]
```

```
        # Calculate distances to other vertices
        distances = []
        for key2, u in vertices.items():
            if u[0]!=y1 and u[1]!=x1:
                try:
                    distances.append((u, distance_formula(x1, y1, u[1], u[0])))
                except Exception as e:
                    print(f"Error calculating distance: {e}")

        # Sort distances
        distances.sort(key=lambda x: x[1])

        # Select k nearest neighbors and add edges
        for j in range(k):
            u, _ = distances[j]
            val = {i for i in martagraph if martagraph[i]==u}
            va = val.pop()
            edges.append((name, va, distances[j][1]))

    return edges

G = nx.Graph()

# Step 3: Add nodes and edges
for name, coordinates in martagraph.items():
    G.add_node(name, pos=coordinates)  # Add node with 'pos' attribute

edges = knn_edges(160, 8, martagraph)
#print(edges)
G.add_weighted_edges_from(edges)

# Plot the graph
pos = nx.get_node_attributes(G, 'pos')
nx.draw(G, pos, with_labels=False, node_size=10)
plt.show()
```



In [ ]:
```python
for k in range(20):
    G2 = nx.Graph()

    # Step 3: Add nodes and edges
    for name, coordinates in martagraph.items():
        G2.add_node(name, pos=coordinates)  # Add node with 'pos' attribute

    edges = knn_edges(160, k, martagraph)
    G2.add_weighted_edges_from(edges)
    try:
        eulerian_circuit = list(nx.eulerian_circuit(G))
        # Print the Eulerian circuit
        print("Eulerian Circuit:", eulerian_circuit)

        # Visualize the graph and the Eulerian circuit
        pos = nx.spring_layout(G2)
        nx.draw(G2, pos, with_labels=True, font_weight='bold')
        nx.draw_networkx_edges(G2, pos, edgelist=eulerian_circuit, edge_color='r',
        plt.show()
    except Exception as e:
        print(f"Error not Eulerian: {e}")
```

```
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
Error not Eulerian: G is not Eulerian.
```

In [ ]:
```
'''def hamiltonian(graph, start, path=[]):
    path = path + [start]
    if len(path) == len(graph.nodes):
        return path
    for node in graph.neighbors(start):
        if node not in path:
            new_path = hamiltonian(graph, node, path)
            if new_path:
                return new_path
    return None

starting_node = list(G.nodes)[0]

hamiltonian_path = hamiltonian(graph=G, start=starting_node)

print("Hamiltonian Path:", hamiltonian_path)
'''
# Visualize the graph and the Hamiltonian path
'''
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, font_weight='bold')
nx.draw_networkx_nodes(G, pos, nodelist=hamiltonian_path, node_color='r')
nx.draw_networkx_edges(G, pos, edgelist=[(hamiltonian_path[i], hamiltonian_path[i +
plt.show()
'''
```

Out[ ]:
```
"\npos = nx.spring_layout(G)\nnx.draw(G, pos, with_labels=True, font_weight='bol
d')\nnx.draw_networkx_nodes(G, pos, nodelist=hamiltonian_path, node_color='r')\nn
x.draw_networkx_edges(G, pos, edgelist=[(hamiltonian_path[i], hamiltonian_path[i +
1]) for i in range(len(hamiltonian_path) - 1)], edge_color='r', width=2)\nplt.show
()\n"
```

In [ ]:
```
Ge = nx.euler.eulerize(G)

eulerian_circuit = list(nx.eulerian_circuit(Ge))
```

```
# Print the Eulerian circuit
print("Eulerian Circuit:", eulerian_circuit)

# Visualize the graph and the Eulerian circuit
pos = nx.get_node_attributes(Ge, 'pos')
nx.draw(Ge, pos, with_labels=False, node_size=12)
nx.draw_networkx_edges(Ge, pos, edgelist=eulerian_circuit, edge_color='r', width=1.
plt.show()
```

```
Eulerian Circuit: [('Airport', 'City of South Fulton Station'), ('City of South Fu
lton Station', 'Cascade Springs'), ('Cascade Springs', 'Deerwood Park'), ('Deerwoo
d Park', 'Melvin Drive Park'), ('Melvin Drive Park', 'Cascade Springs'), ('Cascade
Springs', 'Sandtown'), ('Sandtown', 'Melvin Drive Park'), ('Melvin Drive Park', 'S
weetwater Creek State Park'), ('Sweetwater Creek State Park', 'Lions Park'), ('Lio
ns Park', 'Woodrow Willson Park'), ('Woodrow Willson Park', 'Hurt Road Park'), ('H
urt Road Park', 'Lions Park'), ('Lions Park', 'Heritage Park'), ('Heritage Park',
'Bishop Park'), ('Bishop Park', 'Cheatham Hill '), ('Cheatham Hill ', 'Mud Cree
k'), ('Mud Creek', 'Bishop Park'), ('Bishop Park', 'Wild Horse'), ('Wild Horse',
'Woodrow Willson Park'), ('Woodrow Willson Park', 'Heritage Park'), ('Heritage Par
k', 'Standing Peachtree Park'), ('Standing Peachtree Park', 'Palisades Unit'), ('P
alisades Unit', 'Sope Creek '), ('Sope Creek ', 'Morgan Falls'), ('Morgan Falls',
'Gold Branch Trail'), ('Gold Branch Trail', 'East Cobb Park'), ('East Cobb Park',
'Morgan Falls'), ('Morgan Falls', 'Downtown Roswell'), ('Downtown Roswell', 'North
Roswell'), ('North Roswell', 'Hickory Flat'), ('Hickory Flat', 'Hobgood Park'),
('Hobgood Park', 'Tomahawk'), ('Tomahawk', 'North Roswell'), ('North Roswell', 'We
st Field'), ('West Field', 'Morgan Falls'), ('Morgan Falls', 'Horseshoe Bend '),
('Horseshoe Bend ', 'Jones Bridge Park'), ('Jones Bridge Park', 'Horseshoe Bend
'), ('Horseshoe Bend ', 'Gold Branch Trail'), ('Gold Branch Trail', 'Sope Creek
'), ('Sope Creek ', 'East Cobb Park'), ('East Cobb Park', 'Downtown Marietta'),
('Downtown Marietta', 'Bishop Park'), ('Bishop Park', 'Hurt Road Park'), ('Hurt Ro
ad Park', 'Mud Creek'), ('Mud Creek', 'Wild Horse'), ('Wild Horse', 'Hurt Road Par
k'), ('Hurt Road Park', 'Silver Comet Trail'), ('Silver Comet Trail', 'Palisades U
nit'), ('Palisades Unit', 'West Buckhead'), ('West Buckhead', 'Standing Peachtree
Park'), ('Standing Peachtree Park', 'Silver Comet Trail'), ('Silver Comet Trail',
'Lions Park'), ('Lions Park', 'Smyrna'), ('Smyrna', 'Lions Park'), ('Lions Park',
'Old Clarkdale Park'), ('Old Clarkdale Park', 'Mud Creek'), ('Mud Creek', 'Oregon
Park'), ('Oregon Park', 'Allatoona Creek Park'), ('Allatoona Creek Park', 'Cheatha
m Hill '), ('Cheatham Hill ', 'Downtown Marietta'), ('Downtown Marietta', 'East Ma
rietta'), ('East Marietta', 'Sope Creek '), ('Sope Creek ', 'Wheeler'), ('Wheele
r', 'Downtown Marietta'), ('Downtown Marietta', 'East Marietta'), ('East Mariett
a', 'East Cobb Park'), ('East Cobb Park', 'Wheeler'), ('Wheeler', 'Sandy Plains'),
('Sandy Plains', 'Hobgood Park'), ('Hobgood Park', 'Noonday'), ('Noonday', 'Hickor
y Flat'), ('Hickory Flat', 'Tomahawk'), ('Tomahawk', 'Noonday'), ('Noonday', 'East
Cobb Park'), ('East Cobb Park', 'West Field'), ('West Field', 'Gold Branch Trai
l'), ('Gold Branch Trail', 'Downtown Roswell'), ('Downtown Roswell', 'Horseshoe Be
nd '), ('Horseshoe Bend ', 'Peachtree Corners'), ('Peachtree Corners', 'Gwinnett V
illage/GACS'), ('Gwinnett Village/GACS', 'Mountain Park'), ('Mountain Park', 'Bris
coe Park'), ('Briscoe Park', 'Yellow River Park'), ('Yellow River Park', 'Collinsv
ille'), ('Collinsville', 'Flat Rock'), ('Flat Rock', 'Village Park'), ('Village Pa
rk', 'Kennedy Memorial Gardens'), ('Kennedy Memorial Gardens', 'Lake Charlotte Pre
serve'), ('Lake Charlotte Preserve', 'Browns Mill Park'), ('Browns Mill Park', 'Ke
nnedy Memorial Gardens'), ('Kennedy Memorial Gardens', 'Chapel Hill Park'), ('Chap
el Hill Park', 'Village Park'), ('Village Park', 'Exchange Park'), ('Exchange Par
k', 'Kennedy Memorial Gardens'), ('Kennedy Memorial Gardens', 'West Stonecrest'),
('West Stonecrest', 'Exchange Park'), ('Exchange Park', 'Chapel Hill Park'), ('Cha
pel Hill Park', 'Flat Rock'), ('Flat Rock', 'West Stonecrest'), ('West Stonecres
t', 'Collinsville'), ('Collinsville', 'East Stonecrest'), ('East Stonecrest', 'Wes
t Stonecrest'), ('West Stonecrest', 'Chapel Hill Park'), ('Chapel Hill Park', 'Ara
bia Mountain'), ('Arabia Mountain', 'Village Park'), ('Village Park', 'Lake Charlo
tte Preserve'), ('Lake Charlotte Preserve', 'Browns Mill Park'), ('Browns Mill Par
k', 'Lake City'), ('Lake City', 'Lake Charlotte Preserve'), ('Lake Charlotte Prese
rve', 'Forest Park'), ('Forest Park', 'Village Park'), ('Village Park', 'Lake Cit
y'), ('Lake City', 'Kennedy Memorial Gardens'), ('Kennedy Memorial Gardens', 'The
Meadows'), ('The Meadows', 'Collinsville'), ('Collinsville', 'Arabia Mountain'),
('Arabia Mountain', 'East Stonecrest'), ('East Stonecrest', 'Flat Rock'), ('Flat R
```

ock', 'The Meadows'), ('The Meadows', 'Chapel Hill Park'), ('Chapel Hill Park', 'R
edan'), ('Redan', 'Exchange Park'), ('Exchange Park', 'The Meadows'), ('The Meadow
s', 'East Stonecrest'), ('East Stonecrest', 'Yellow River Park'), ('Yellow River P
ark', 'Mountain Park'), ('Mountain Park', 'Bryson Park'), ('Bryson Park', 'McDanie
l Farm'), ('McDaniel Farm', 'Gwinnett Village/GACS'), ('Gwinnett Village/GACS', 'M
ountain Park'), ('Mountain Park', 'West Snellville'), ('West Snellville', 'Yellow
River Park'), ('Yellow River Park', 'Redan'), ('Redan', 'Yellow River Park'), ('Ye
llow River Park', 'Central Snellville'), ('Central Snellville', 'Mountain Park'),
('Mountain Park', 'Stone Mountain'), ('Stone Mountain', 'Yellow River Park'), ('Ye
llow River Park', 'Lithonia'), ('Lithonia', 'Flat Rock'), ('Flat Rock', 'Arabia Mo
untain'), ('Arabia Mountain', 'West Stonecrest'), ('West Stonecrest', 'The Meadow
s'), ('The Meadows', 'Arabia Mountain'), ('Arabia Mountain', 'Redan'), ('Redan',
'Arabia Mountain'), ('Arabia Mountain', 'Lithonia'), ('Lithonia', 'Collinsville'),
('Collinsville', 'Redan'), ('Redan', 'East Stonecrest'), ('East Stonecrest', 'Lith
onia'), ('Lithonia', 'West Stonecrest'), ('West Stonecrest', 'Redan'), ('Redan',
'Lithonia'), ('Lithonia', 'The Meadows'), ('The Meadows', 'Kensington'), ('Kensing
ton', 'Exchange Park'), ('Exchange Park', 'Avondale'), ('Avondale', 'Exchange Par
k'), ('Exchange Park', 'Indian Creek'), ('Indian Creek', 'The Meadows'), ('The Mea
dows', 'Redan'), ('Redan', 'Stone Mountain'), ('Stone Mountain', 'Northlake'), ('N
orthlake', 'North Druid Hills'), ('North Druid Hills', 'Emory'), ('Emory', 'Northl
ake'), ('Northlake', 'Bryson Park'), ('Bryson Park', 'Gwinnett Village/GACS'), ('G
winnett Village/GACS', 'Berkeley Lake'), ('Berkeley Lake', 'Jones Bridge Park'),
('Jones Bridge Park', 'Peachtree Corners'), ('Peachtree Corners', 'McDaniel Far
m'), ('McDaniel Farm', 'Sugarloaf'), ('Sugarloaf', 'Lawrenceville'), ('Lawrencevil
le', 'Collins Hill Park'), ('Collins Hill Park', 'Lawrenceville'), ('Lawrencevill
e', 'Gwinnett Airport'), ('Gwinnett Airport', 'Sugarloaf'), ('Sugarloaf', 'Collins
Hill Park'), ('Collins Hill Park', 'South Buford'), ('South Buford', 'Buford Excha
nge'), ('Buford Exchange', 'Gwinnett Airport'), ('Gwinnett Airport', 'Mall of Geor
gia'), ('Mall of Georgia', 'Gwinnett Airport'), ('Gwinnett Airport', 'Collins Hill
Park'), ('Collins Hill Park', 'Buford Exchange'), ('Buford Exchange', 'Mall of Geo
rgia'), ('Mall of Georgia', 'Collins Hill Park'), ('Collins Hill Park', 'Dacula'),
('Dacula', 'Buford Exchange'), ('Buford Exchange', 'Lawrenceville'), ('Lawrencevil
le', 'North Snellville'), ('North Snellville', 'Briscoe Park'), ('Briscoe Park',
'Grayson'), ('Grayson', 'Gwinnett Airport'), ('Gwinnett Airport', 'North Snellvill
e'), ('North Snellville', 'Grayson'), ('Grayson', 'West Snellville'), ('West Snell
ville', 'Briscoe Park'), ('Briscoe Park', 'Lilburn'), ('Lilburn', 'West Snellvill
e'), ('West Snellville', 'North Snellville'), ('North Snellville', 'Dacula'), ('Da
cula', 'Grayson'), ('Grayson', 'Lawrenceville'), ('Lawrenceville', 'Dacula'), ('Da
cula', 'Grayson'), ('Grayson', 'Central Snellville'), ('Central Snellville', 'Nort
h Snellville'), ('North Snellville', 'Lilburn'), ('Lilburn', 'Gwinnett Village/GAC
S'), ('Gwinnett Village/GACS', 'Tucker'), ('Tucker', 'Mountain Park'), ('Mountain
Park', 'Lilburn'), ('Lilburn', 'McDaniel Farm'), ('McDaniel Farm', 'Berkeley Lak
e'), ('Berkeley Lake', 'Bryson Park'), ('Bryson Park', 'Lilburn'), ('Lilburn', 'We
st Lawrenceville'), ('West Lawrenceville', 'Lilburn'), ('Lilburn', 'Central Snellv
ille'), ('Central Snellville', 'Briscoe Park'), ('Briscoe Park', 'Loganville'),
('Loganville', 'Dacula'), ('Dacula', 'Mall of Georgia'), ('Mall of Georgia', 'Sout
h Buford'), ('South Buford', 'Big Creek'), ('Big Creek', 'Fowler Park'), ('Fowler
Park', 'Bell Memorial Park'), ('Bell Memorial Park', 'Tomahawk'), ('Tomahawk', 'We
st Field'), ('West Field', 'Downtown Roswell'), ('Downtown Roswell', 'Alpharett
a'), ('Alpharetta', 'Fowler Park'), ('Fowler Park', 'Cumming'), ('Cumming', 'Bell
Memorial Park'), ('Bell Memorial Park', 'North Roswell'), ('North Roswell', 'Alpha
retta'), ('Alpharetta', 'Bell Memorial Park'), ('Bell Memorial Park', 'Big Cree
k'), ('Big Creek', 'Cumming'), ('Cumming', 'East Johns Creek'), ('East Johns Cree
k', 'Big Creek'), ('Big Creek', 'Sugar Hill'), ('Sugar Hill', 'Cumming'), ('Cummin
g', 'Buford'), ('Buford', 'Big Creek'), ('Big Creek', 'Suwanee'), ('Suwanee', 'Cum
ming'), ('Cumming', 'South Buford'), ('South Buford', 'East Johns Creek'), ('East

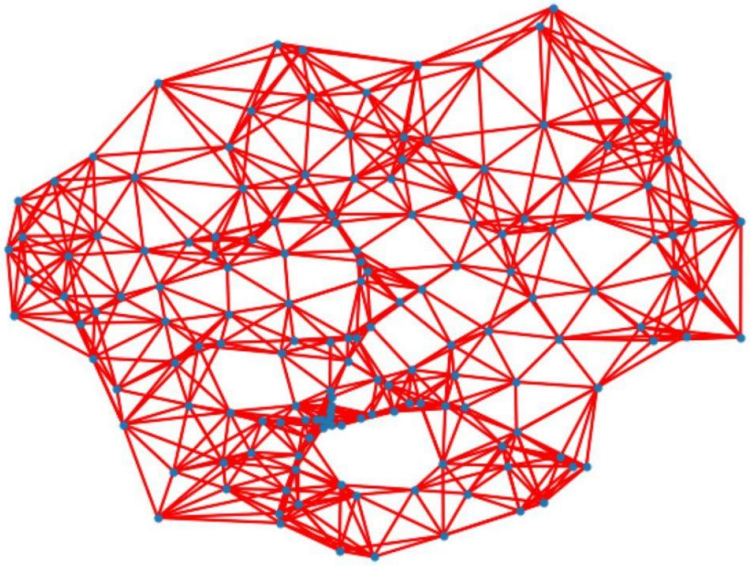
```
Johns Creek', 'Fowler Park'), ('Fowler Park', 'Johns Creek'), ('Johns Creek', 'Alp
haretta'), ('Alpharetta', 'Jones Bridge Park'), ('Jones Bridge Park', 'North Poin
t'), ('North Point', 'Bell Memorial Park'), ('Bell Memorial Park', 'Hickory Fla
t'), ('Hickory Flat', 'Woodstock'), ('Woodstock', 'North Roswell'), ('North Roswel
l', 'North Point'), ('North Point', 'Downtown Roswell'), ('Downtown Roswell', 'Eas
t Cobb'), ('East Cobb', 'Wheeler'), ('Wheeler', 'East Marietta'), ('East Mariett
a', 'Cheatham Hill '), ('Cheatham Hill ', 'Oregon Park'), ('Oregon Park', 'Roxan
a'), ('Roxana', 'Allatoona Creek Park'), ('Allatoona Creek Park', 'Wildwood'), ('W
ildwood', 'Oregon Park'), ('Oregon Park', 'Mt Tabor Park'), ('Mt Tabor Park', 'Wil
dwood'), ('Wildwood', 'Roxana'), ('Roxana', 'Mt Tabor Park'), ('Mt Tabor Park', 'A
llatoona Creek Park'), ('Allatoona Creek Park', 'Lost Mountain Park'), ('Lost Moun
tain Park', 'Wildwood'), ('Wildwood', 'Powder Creek Crossing'), ('Powder Creek Cro
ssing', 'Oregon Park'), ('Oregon Park', 'Lake Lucile'), ('Lake Lucile', 'Roxana'),
('Roxana', 'Powder Creek Crossing'), ('Powder Creek Crossing', 'Roxana'), ('Roxan
a', 'Lost Mountain Park'), ('Lost Mountain Park', 'Mud Creek'), ('Mud Creek', 'Lak
e Lucile'), ('Lake Lucile', 'Mt Tabor Park'), ('Mt Tabor Park', 'Powder Creek Cros
sing'), ('Powder Creek Crossing', 'Wild Horse'), ('Wild Horse', 'Lake Lucile'),
('Lake Lucile', 'Powder Creek Crossing'), ('Powder Creek Crossing', 'Lost Mountain
Park'), ('Lost Mountain Park', 'Oregon Park'), ('Oregon Park', 'Lost Mountain Par
k'), ('Lost Mountain Park', 'Wild Horse'), ('Wild Horse', 'Old Clarkdale Park'),
('Old Clarkdale Park', 'Sweetwater Creek State Park'), ('Sweetwater Creek State Pa
rk', 'Old Clarkdale Park'), ('Old Clarkdale Park', 'Hurt Road Park'), ('Hurt Road
Park', 'Heritage Park'), ('Heritage Park', 'Silver Comet Trail'), ('Silver Comet T
rail', 'Westminister'), ('Westminister', 'Palisades Unit'), ('Palisades Unit', 'Ma
rietta/Lockheed'), ('Marietta/Lockheed', 'Sope Creek '), ('Sope Creek ', 'Kennesaw
State Univ'), ('Kennesaw State Univ', 'East Cobb Park'), ('East Cobb Park', 'Sandy
Plains'), ('Sandy Plains', 'West Field'), ('West Field', 'Noonday'), ('Noonday',
'Sandy Plains'), ('Sandy Plains', 'Noonday'), ('Noonday', 'Acworth'), ('Acworth',
'Lost Mountain Park'), ('Lost Mountain Park', 'Cheatham Hill '), ('Cheatham Hill
', 'Kennesaw State Univ'), ('Kennesaw State Univ', 'Bishop Park'), ('Bishop Park',
'Kennesaw State Univ'), ('Kennesaw State Univ', 'Sandy Plains'), ('Sandy Plains',
'East Cobb'), ('East Cobb', 'Gold Branch Trail'), ('Gold Branch Trail', 'Sandy Spr
ings'), ('Sandy Springs', 'Sope Creek '), ('Sope Creek ', 'North Springs'), ('Nort
h Springs', 'Gold Branch Trail'), ('Gold Branch Trail', 'Dunwoody'), ('Dunwoody',
'Gold Branch Trail'), ('Gold Branch Trail', 'Roswell'), ('Roswell', 'Jones Bridge
Park'), ('Jones Bridge Park', 'Johns Creek'), ('Johns Creek', 'Peachtree Corner
s'), ('Peachtree Corners', 'Berkeley Lake'), ('Berkeley Lake', 'Sugarloaf'), ('Sug
arloaf', 'West Lawrenceville'), ('West Lawrenceville', 'Grayson'), ('Grayson', 'Lo
ganville'), ('Loganville', 'Central Snellville'), ('Central Snellville', 'West Sne
llville'), ('West Snellville', 'Loganville'), ('Loganville', 'North Snellville'),
('North Snellville', 'West Lawrenceville'), ('West Lawrenceville', 'Dacula'), ('Da
cula', 'Gwinnett Airport'), ('Gwinnett Airport', 'West Lawrenceville'), ('West Law
renceville', 'Loganville'), ('Loganville', 'Lawrenceville'), ('Lawrenceville', 'We
st Lawrenceville'), ('West Lawrenceville', 'Collins Hill Park'), ('Collins Hill Pa
rk', 'Sugar Hill'), ('Sugar Hill', 'East Johns Creek'), ('East Johns Creek', 'Bufo
rd'), ('Buford', 'Buford Exchange'), ('Buford Exchange', 'Sugar Hill'), ('Sugar Hi
ll', 'Mall of Georgia'), ('Mall of Georgia', 'Buford'), ('Buford', 'Sugar Hill'),
('Sugar Hill', 'South Buford'), ('South Buford', 'Buford'), ('Buford', 'Suwanee'),
('Suwanee', 'Mall of Georgia'), ('Mall of Georgia', 'East Duluth'), ('East Dulut
h', 'South Buford'), ('South Buford', 'Suwanee'), ('Suwanee', 'East Johns Creek'),
('East Johns Creek', 'Johns Creek'), ('Johns Creek', 'Berkeley Lake'), ('Berkeley
Lake', 'East Duluth'), ('East Duluth', 'Buford Exchange'), ('Buford Exchange', 'Su
wanee'), ('Suwanee', 'Collins Hill Park'), ('Collins Hill Park', 'East Duluth'),
('East Duluth', 'Sugar Hill'), ('Sugar Hill', 'Suwanee'), ('Suwanee', 'East Johns
Creek'), ('East Johns Creek', 'East Duluth'), ('East Duluth', 'Suwanee'), ('Suwane
e', 'Sugarloaf'), ('Sugarloaf', 'East Duluth'), ('East Duluth', 'Johns Creek'),
```

```
('Johns Creek', 'Duluth'), ('Duluth', 'Sugarloaf'), ('Sugarloaf', 'Duluth'), ('Dul
uth', 'East Johns Creek'), ('East Johns Creek', 'Avalon'), ('Avalon', 'Fowler Par
k'), ('Fowler Park', 'Milton'), ('Milton', 'Bell Memorial Park'), ('Bell Memorial
Park', 'Holy Springs'), ('Holy Springs', 'North Roswell'), ('North Roswell', 'Rosw
ell'), ('Roswell', 'Alpharetta'), ('Alpharetta', 'North Point'), ('North Point',
'Horseshoe Bend '), ('Horseshoe Bend ', 'Roswell'), ('Roswell', 'Downtown Roswel
l'), ('Downtown Roswell', 'Avalon'), ('Avalon', 'Bell Memorial Park'), ('Bell Memo
rial Park', 'Milton'), ('Milton', 'Alpharetta'), ('Alpharetta', 'Tomahawk'), ('Tom
ahawk', 'Woodstock'), ('Woodstock', 'Sandy Plains'), ('Sandy Plains', 'Downtown Ma
rietta'), ('Downtown Marietta', 'Kennesaw State Univ'), ('Kennesaw State Univ', 'W
heeler'), ('Wheeler', 'Marietta/Lockheed'), ('Marietta/Lockheed', 'East Cobb Par
k'), ('East Cobb Park', 'East Cobb'), ('East Cobb', 'Noonday'), ('Noonday', 'Woods
tock'), ('Woodstock', 'Hobgood Park'), ('Hobgood Park', 'Holy Springs'), ('Holy Sp
rings', 'Woodstock'), ('Woodstock', 'West Field'), ('West Field', 'East Cobb'),
('East Cobb', 'Morgan Falls'), ('Morgan Falls', 'Sandy Springs'), ('Sandy Spring
s', 'North Springs'), ('North Springs', 'Horseshoe Bend '), ('Horseshoe Bend ', 'N
orcross'), ('Norcross', 'Peachtree Corners'), ('Peachtree Corners', 'Duluth'), ('D
uluth', 'East Duluth'), ('East Duluth', 'McDaniel Farm'), ('McDaniel Farm', 'Dulut
h'), ('Duluth', 'Gwinnett Village/GACS'), ('Gwinnett Village/GACS', 'Norcross'),
('Norcross', 'Berkeley Lake'), ('Berkeley Lake', 'Duluth'), ('Duluth', 'Jones Brid
ge Park'), ('Jones Bridge Park', 'Avalon'), ('Avalon', 'Johns Creek'), ('Johns Cre
ek', 'North Point'), ('North Point', 'Roswell'), ('Roswell', 'Avalon'), ('Avalon',
'North Roswell'), ('North Roswell', 'Milton'), ('Milton', 'Hickory Flat'), ('Hicko
ry Flat', 'Holy Springs'), ('Holy Springs', 'Tomahawk'), ('Tomahawk', 'Milton'),
('Milton', 'North Point'), ('North Point', 'Avalon'), ('Avalon', 'Alpharetta'),
('Alpharetta', 'Avalon'), ('Avalon', 'Milton'), ('Milton', 'Holy Springs'), ('Holy
Springs', 'Noonday'), ('Noonday', 'Kennesaw'), ('Kennesaw', 'Sandy Plains'), ('San
dy Plains', 'East Marietta'), ('East Marietta', 'Marietta/Lockheed'), ('Marietta/L
ockheed', 'Downtown Marietta'), ('Downtown Marietta', 'Kennesaw'), ('Kennesaw', 'E
ast Marietta'), ('East Marietta', 'Kennesaw State Univ'), ('Kennesaw State Univ',
'Marietta/Lockheed'), ('Marietta/Lockheed', 'Bishop Park'), ('Bishop Park', 'Smyrn
a'), ('Smyrna', 'Standing Peachtree Park'), ('Standing Peachtree Park', 'Westminis
ter'), ('Westminister', 'Smyrna'), ('Smyrna', 'Hurt Road Park'), ('Hurt Road Par
k', 'Powder Springs'), ('Powder Springs', 'Mud Creek'), ('Mud Creek', 'Powder Spri
ngs'), ('Powder Springs', 'Lost Mountain Park'), ('Lost Mountain Park', 'Lake Luci
le'), ('Lake Lucile', 'Old Clarkdale Park'), ('Old Clarkdale Park', 'Woodrow Wills
on Park'), ('Woodrow Willson Park', 'Sweetwater Creek State Park'), ('Sweetwater C
reek State Park', 'Sandtown'), ('Sandtown', 'Deerwood Park'), ('Deerwood Park', 'C
ity of South Fulton Station'), ('City of South Fulton Station', 'Sweetwater Creek
State Park'), ('Sweetwater Creek State Park', 'Heritage Park'), ('Heritage Park',
'Adamsville'), ('Adamsville', 'Sandtown'), ('Sandtown', 'City of South Fulton Stat
ion'), ('City of South Fulton Station', 'Melvin Drive Park'), ('Melvin Drive Par
k', 'Six Flags Over GA'), ('Six Flags Over GA', 'Heritage Park'), ('Heritage Par
k', 'Smyrna'), ('Smyrna', 'Marietta/Lockheed'), ('Marietta/Lockheed', 'Cheatham Hi
ll '), ('Cheatham Hill ', 'Kennesaw'), ('Kennesaw', 'Hobgood Park'), ('Hobgood Par
k', 'Acworth'), ('Acworth', 'Oregon Park'), ('Oregon Park', 'Kennesaw'), ('Kennesa
w', 'Wildwood'), ('Wildwood', 'Acworth'), ('Acworth', 'Allatoona Creek Park'), ('A
llatoona Creek Park', 'Kennesaw'), ('Kennesaw', 'Acworth'), ('Acworth', 'Roxana'),
('Roxana', 'Hiram'), ('Hiram', 'Lost Mountain Park'), ('Lost Mountain Park', 'Mt T
abor Park'), ('Mt Tabor Park', 'Hiram'), ('Hiram', 'Old Clarkdale Park'), ('Old Cl
arkdale Park', 'Powder Springs'), ('Powder Springs', 'Lake Lucile'), ('Lake Lucil
e', 'Hiram'), ('Hiram', 'Wild Horse'), ('Wild Horse', 'Powder Springs'), ('Powder
Springs', 'Powder Creek Crossing'), ('Powder Creek Crossing', 'Hiram'), ('Hiram',
'Powder Springs'), ('Powder Springs', 'Woodrow Willson Park'), ('Woodrow Willson P
ark', 'Six Flags Over GA'), ('Six Flags Over GA', 'Melvin Drive Park'), ('Melvin D
rive Park', 'Oakland City'), ('Oakland City', 'Lake Charlotte Preserve'), ('Lake C
```

harlotte Preserve', 'Hapeville'), ('Hapeville', 'Lake City'), ('Lake City', 'Fores
t Park'), ('Forest Park', 'Browns Mill Park'), ('Browns Mill Park', 'West End'),
('West End', 'West Lake'), ('West Lake', 'Cascade Springs'), ('Cascade Springs',
'Oakland City'), ('Oakland City', 'Vine City'), ('Vine City', 'Peachtree Center'),
('Peachtree Center', 'North Avenue'), ('North Avenue', 'Vine City'), ('Vine City',
'West End'), ('West End', 'Oakland City'), ('Oakland City', 'Browns Mill Park'),
('Browns Mill Park', 'Hapeville'), ('Hapeville', 'Deerwood Park'), ('Deerwood Par
k', 'Lakewood/Fort McPherson'), ('Lakewood/Fort McPherson', 'Melvin Drive Park'),
('Melvin Drive Park', 'Adamsville'), ('Adamsville', 'Sweetwater Creek State Par
k'), ('Sweetwater Creek State Park', 'Six Flags Over GA'), ('Six Flags Over GA',
'Sandtown'), ('Sandtown', 'H. E. Holmes'), ('H. E. Holmes', 'Melvin Drive Park'),
('Melvin Drive Park', 'East Point'), ('East Point', 'Lake Charlotte Preserve'),
('Lake Charlotte Preserve', 'Lakewood/Fort McPherson'), ('Lakewood/Fort McPherso
n', 'Cascade Springs'), ('Cascade Springs', 'Adamsville'), ('Adamsville', 'Lions P
ark'), ('Lions Park', 'Six Flags Over GA'), ('Six Flags Over GA', 'Silver Comet Tr
ail'), ('Silver Comet Trail', 'Smyrna'), ('Smyrna', 'Palisades Unit'), ('Palisades
Unit', 'Medical Center'), ('Medical Center', 'North Springs'), ('North Springs',
'Morgan Falls'), ('Morgan Falls', 'Dunwoody'), ('Dunwoody', 'Norcross'), ('Norcros
s', 'Bryson Park'), ('Bryson Park', 'Tucker'), ('Tucker', 'Stone Mountain'), ('Sto
ne Mountain', 'Kensington'), ('Kensington', 'Northlake'), ('Northlake', 'Tucker'),
('Tucker', 'North Druid Hills'), ('North Druid Hills', 'CDC'), ('CDC', 'Emory'),
('Emory', 'Clarkston'), ('Clarkston', 'Tucker'), ('Tucker', 'Doraville'), ('Doravi
lle', 'North Springs'), ('North Springs', 'Dunwoody'), ('Dunwoody', 'Sandy Spring
s'), ('Sandy Springs', 'Medical Center'), ('Medical Center', 'Palisades Unit'),
('Palisades Unit', 'Buckhead'), ('Buckhead', 'Standing Peachtree Park'), ('Standin
g Peachtree Park', 'Lindbergh Center'), ('Lindbergh Center', 'Emory'), ('Emory',
'Kensington'), ('Kensington', 'Redan'), ('Redan', 'Indian Creek'), ('Indian Cree
k', 'Stone Mountain'), ('Stone Mountain', 'Clarkston'), ('Clarkston', 'North Druid
Hills'), ('North Druid Hills', 'Kensington'), ('Kensington', 'Northlake'), ('North
lake', 'Doraville'), ('Doraville', 'North Springs'), ('North Springs', 'Chamble
e'), ('Chamblee', 'Northlake'), ('Northlake', 'Clarkston'), ('Clarkston', 'Indian
Creek'), ('Indian Creek', 'Clarkston'), ('Clarkston', 'Kensington'), ('Kensingto
n', 'Indian Creek'), ('Indian Creek', 'Decatur'), ('Decatur', 'Clarkston'), ('Clar
kston', 'Avondale'), ('Avondale', 'Indian Creek'), ('Indian Creek', 'East Lake'),
('East Lake', 'Kensington'), ('Kensington', 'Edgewood/Candler Park'), ('Edgewood/C
andler Park', 'CDC'), ('CDC', 'East Lake'), ('East Lake', 'North Druid Hills'),
('North Druid Hills', 'Decatur'), ('Decatur', 'CDC'), ('CDC', 'Midtown'), ('Midtow
n', 'Vine City'), ('Vine City', 'West Lake'), ('West Lake', 'Adamsville'), ('Adams
ville', 'Six Flags Over GA'), ('Six Flags Over GA', 'H. E. Holmes'), ('H. E. Holme
s', 'Oakland City'), ('Oakland City', 'West Lake'), ('West Lake', 'Garnett'), ('Ga
rnett', 'Oakland City'), ('Oakland City', 'Hapeville'), ('Hapeville', 'Forest Par
k'), ('Forest Park', 'East Point'), ('East Point', 'Cascade Springs'), ('Cascade S
prings', 'H. E. Holmes'), ('H. E. Holmes', 'West End'), ('West End', 'Adamsvill
e'), ('Adamsville', 'H. E. Holmes'), ('H. E. Holmes', 'Vine City'), ('Vine City',
'King Memorial'), ('King Memorial', 'Peachtree Center'), ('Peachtree Center', 'Mid
town'), ('Midtown', 'North Avenue'), ('North Avenue', 'Inman Park/Reynoldstown'),
('Inman Park/Reynoldstown', 'Emory'), ('Emory', 'Lenox'), ('Lenox', 'Medical Cente
r'), ('Medical Center', 'Dunwoody'), ('Dunwoody', 'Doraville'), ('Doraville', 'San
dy Springs'), ('Sandy Springs', 'Chamblee'), ('Chamblee', 'Norcross'), ('Norcros
s', 'Doraville'), ('Doraville', 'Medical Center'), ('Medical Center', 'Chamblee'),
('Chamblee', 'Dunwoody'), ('Dunwoody', 'Brookhaven/Oglethorpe'), ('Brookhaven/Ogle
thorpe', 'Doraville'), ('Doraville', 'Chamblee'), ('Chamblee', 'Buckhead'), ('Buck
head', 'CDC'), ('CDC', 'Lindbergh Center'), ('Lindbergh Center', 'Westminister'),
('Westminister', 'West Buckhead'), ('West Buckhead', 'Lindbergh Center'), ('Lindbe
rgh Center', 'Midtown'), ('Midtown', 'Georgia State'), ('Georgia State', 'West En
d'), ('West End', 'Lakewood/Fort McPherson'), ('Lakewood/Fort McPherson', 'Browns

```
Mill Park'), ('Browns Mill Park', 'East Point'), ('East Point', 'City of South Ful
ton Station'), ('City of South Fulton Station', 'College Park'), ('College Park',
'Lake Charlotte Preserve'), ('Lake Charlotte Preserve', 'Airport'), ('Airport', 'L
ake City'), ('Lake City', 'College Park'), ('College Park', 'Oakland City'), ('Oak
land City', 'Lakewood/Fort McPherson'), ('Lakewood/Fort McPherson', 'Garnett'),
('Garnett', 'West Lake'), ('West Lake', 'Five Points'), ('Five Points', 'Midtow
n'), ('Midtown', 'Five Points'), ('Five Points', 'West End'), ('West End', 'Garnet
t'), ('Garnett', 'Vine City'), ('Vine City', 'Georgia State'), ('Georgia State',
'Inman Park/Reynoldstown'), ('Inman Park/Reynoldstown', 'Peachtree Center'), ('Pea
chtree Center', 'Georgia State'), ('Georgia State', 'North Avenue'), ('North Avenu
e', 'Five Points'), ('Five Points', 'Inman Park/Reynoldstown'), ('Inman Park/Reyno
ldstown', 'King Memorial'), ('King Memorial', 'Georgia State'), ('Georgia State',
'Garnett'), ('Garnett', 'King Memorial'), ('King Memorial', 'Five Points'), ('Five
Points', 'Garnett'), ('Garnett', 'Peachtree Center'), ('Peachtree Center', 'Five P
oints'), ('Five Points', 'Georgia State'), ('Georgia State', 'Edgewood/Candler Par
k'), ('Edgewood/Candler Park', 'Emory'), ('Emory', 'East Lake'), ('East Lake', 'Ki
ng Memorial'), ('King Memorial', 'Edgewood/Candler Park'), ('Edgewood/Candler Par
k', 'Inman Park/Reynoldstown'), ('Inman Park/Reynoldstown', 'East Lake'), ('East L
ake', 'Edgewood/Candler Park'), ('Edgewood/Candler Park', 'Decatur'), ('Decatur',
'Inman Park/Reynoldstown'), ('Inman Park/Reynoldstown', 'Civic Center'), ('Civic C
enter', 'Edgewood/Candler Park'), ('Edgewood/Candler Park', 'North Avenue'), ('Nor
th Avenue', 'GWCC/CNN Center'), ('GWCC/CNN Center', 'King Memorial'), ('King Memor
ial', 'Civic Center'), ('Civic Center', 'Georgia State'), ('Georgia State', 'GWCC/
CNN Center'), ('GWCC/CNN Center', 'West Lake'), ('West Lake', 'H. E. Holmes'),
('H. E. Holmes', 'Ashby'), ('Ashby', 'Adamsville'), ('Adamsville', 'Bankhead'),
('Bankhead', 'Standing Peachtree Park'), ('Standing Peachtree Park', 'Arts Cente
r'), ('Arts Center', 'CDC'), ('CDC', 'Lenox'), ('Lenox', 'Westminister'), ('Westmi
nister', 'Buckhead'), ('Buckhead', 'Lindbergh Center'), ('Lindbergh Center', 'Leno
x'), ('Lenox', 'West Buckhead'), ('West Buckhead', 'Buckhead'), ('Buckhead', 'Medi
cal Center'), ('Medical Center', 'Brookhaven/Oglethorpe'), ('Brookhaven/Oglethorp
e', 'Lindbergh Center'), ('Lindbergh Center', 'Arts Center'), ('Arts Center', 'Wes
t Buckhead'), ('West Buckhead', 'Brookhaven/Oglethorpe'), ('Brookhaven/Oglethorp
e', 'Chamblee'), ('Chamblee', 'Lenox'), ('Lenox', 'Buckhead'), ('Buckhead', 'Brook
haven/Oglethorpe'), ('Brookhaven/Oglethorpe', 'Lenox'), ('Lenox', 'Brookhaven/Ogle
thorpe'), ('Brookhaven/Oglethorpe', 'North Druid Hills'), ('North Druid Hills', 'A
vondale'), ('Avondale', 'Emory'), ('Emory', 'Decatur'), ('Decatur', 'Kensington'),
('Kensington', 'Avondale'), ('Avondale', 'East Lake'), ('East Lake', 'Decatur'),
('Decatur', 'Avondale'), ('Avondale', 'Edgewood/Candler Park'), ('Edgewood/Candler
Park', 'Peachtree Center'), ('Peachtree Center', 'GWCC/CNN Center'), ('GWCC/CNN Ce
nter', 'Five Points'), ('Five Points', 'Vine City'), ('Vine City', 'GWCC/CNN Cente
r'), ('GWCC/CNN Center', 'Midtown'), ('Midtown', 'Civic Center'), ('Civic Center',
'GWCC/CNN Center'), ('GWCC/CNN Center', 'West End'), ('West End', 'Ashby'), ('Ashb
y', 'H. E. Holmes'), ('H. E. Holmes', 'Bankhead'), ('Bankhead', 'Peachtree Cente
r'), ('Peachtree Center', 'Civic Center'), ('Civic Center', 'Vine City'), ('Vine C
ity', 'Bankhead'), ('Bankhead', 'North Avenue'), ('North Avenue', 'Civic Center'),
('Civic Center', 'Five Points'), ('Five Points', 'Ashby'), ('Ashby', 'West Lake'),
('West Lake', 'Bankhead'), ('Bankhead', 'GWCC/CNN Center'), ('GWCC/CNN Center', 'G
arnett'), ('Garnett', 'Ashby'), ('Ashby', 'Peachtree Center'), ('Peachtree Cente
r', 'Arts Center'), ('Arts Center', 'GWCC/CNN Center'), ('GWCC/CNN Center', 'Ashb
y'), ('Ashby', 'Bankhead'), ('Bankhead', 'Civic Center'), ('Civic Center', 'Arts C
enter'), ('Arts Center', 'North Avenue'), ('North Avenue', 'Arts Center'), ('Arts
Center', 'Bankhead'), ('Bankhead', 'Midtown'), ('Midtown', 'Arts Center'), ('Arts
Center', 'Vine City'), ('Vine City', 'Ashby'), ('Ashby', 'Oakland City'), ('Oaklan
d City', 'East Point'), ('East Point', 'Lakewood/Fort McPherson'), ('Lakewood/Fort
McPherson', 'Hapeville'), ('Hapeville', 'East Point'), ('East Point', 'Deerwood Pa
rk'), ('Deerwood Park', 'College Park'), ('College Park', 'Browns Mill Park'), ('B
```

```
rowns Mill Park', 'Airport'), ('Airport', 'Deerwood Park'), ('Deerwood Park', 'Col
lege Park'), ('College Park', 'Forest Park'), ('Forest Park', 'Airport'), ('Airpor
t', 'Lakewood/Fort McPherson'), ('Lakewood/Fort McPherson', 'College Park'), ('Col
lege Park', 'East Point'), ('East Point', 'Airport'), ('Airport', 'Hapeville'),
('Hapeville', 'College Park'), ('College Park', 'Airport')]
```



In [ ]:
```python
# example path provided by app from Avalon to Airport station using weighted Dijkst
lp = True
while lp:
    try:
        start = input("Enter your starting location: ")
        end = input("Enter your destination: ")
        print(nx.dijkstra_path(G, start, end))
        lp = False
    except Exception as e:
            print(f"Error: {e}")
```

```
Error: Node io not found in graph
['Avalon', 'North Point', 'Horseshoe Bend ', 'North Springs', 'Medical Center', 'B
uckhead', 'Lindbergh Center', 'Arts Center', 'Vine City', 'West End', 'Lakewood/Fo
rt McPherson', 'Airport']
```

In [ ]:
```python
for u, v, data in G.edges(data=True):
    if 'weight' not in data:
        G[u][v]['weight'] = 0.05
    #print(f"Edge ({u}, {v}): {data}")
```

In [ ]:
```python
import torch
import torch.nn as nn
```

```
import torch.optim as optim
import torch_geometric
import tensorflow as tf

# Convert your NetworkX graph to a PyTorch Geometric Data object
node_features = torch.randn(160, 16)
data = torch_geometric.utils.from_networkx(G)
data.x = node_features
print(data)

# Define and train a GNN model
class GNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(GNN, self).__init__()
        self.conv1 = torch_geometric.nn.GraphConv(input_size, hidden_size)
        self.conv2 = torch_geometric.nn.GraphConv(hidden_size, output_size)

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = torch.relu(x)
        x = self.conv2(x, edge_index)
        return x

model = GNN(input_size=16, hidden_size=64, output_size=2)
optimizer = optim.Adam(model.parameters(), lr=0.01)
criterion = nn.CrossEntropyLoss()

# Train the model on your graph data
for epoch in range(20):
    optimizer.zero_grad()
    output = model(data.x, data.edge_index)
    target = torch.ones_like(output)
    loss = criterion(output, target)
    loss.backward()
    optimizer.step()

    print(f'Epoch {epoch + 1}/{20}, Loss: {loss.item()}')

# uses the learned embeddings to guide Hamiltonian path search
node_embeddings = model.conv1(data.x, data.edge_index)
print(node_embeddings)
```

```
Data(edge_index=[2, 1540], pos=[159, 2], weight=[1540], x=[160, 16])
Epoch 1/20, Loss: 3.6754250526428223
Epoch 2/20, Loss: 7.075736045837402
Epoch 3/20, Loss: 4.351600170135498
Epoch 4/20, Loss: 5.848458290100098
Epoch 5/20, Loss: 3.195577621459961
Epoch 6/20, Loss: 3.5237343311309814
Epoch 7/20, Loss: 4.00560998916626
Epoch 8/20, Loss: 2.653991937637329
Epoch 9/20, Loss: 2.611069917678833
Epoch 10/20, Loss: 3.016585350036621
Epoch 11/20, Loss: 2.355003833770752
Epoch 12/20, Loss: 2.118072032928467
Epoch 13/20, Loss: 2.5813546180725098
Epoch 14/20, Loss: 2.4784207344055176
Epoch 15/20, Loss: 1.8085826635360718
Epoch 16/20, Loss: 1.8909556865692139
Epoch 17/20, Loss: 2.1892223358154297
Epoch 18/20, Loss: 1.8698699474334717
Epoch 19/20, Loss: 1.697495460510254
Epoch 20/20, Loss: 1.9151763916015625
tensor([[ 1.2764, -0.0851,  4.0211,  ...,  0.6951, -0.8406, -1.1633],
        [ 0.5514,  0.5048,  1.0761,  ..., -0.4511, -0.9251,  1.0614],
        [ 2.7418, -4.4167,  2.8022,  ..., -1.3299, -2.9913, -3.1469],
        ...,
        [ 0.1665,  1.2033, -1.5127,  ...,  0.8876, -0.6083,  0.1930],
        [ 3.8695,  0.4998,  1.3280,  ..., -0.4611,  1.4111, -2.7711],
        [-0.5052,  0.2717,  1.0724,  ..., -0.5386,  0.3535, -1.3361]],
       grad_fn=<AddBackward0>)
```